PROCEEDINGS OF SCIENCE

Application of the rule-growing algorithm RIPPER to particle physics analysis

Markward Britsch*
Max-Planck-Institut für Kernphysik, PO Box 103980, 69029 Heidelberg, Germany
E-mail: markward@mpi-hd.mpg.de

Nikolai Gagunashvili
University of Akureyri, Borgir, v/Nordurslód, IS-600 Akureyri, Iceland
E-mail: nikolai@unak.is

Michael Schmelling
Max-Planck-Institut fuer Kernphysik, PO Box 103980, 69029 Heidelberg, Germany
E-mail: Michael.Schmelling@mpi-hd.mpg.de

A large hadron machine like the LHC with its high track multiplicities always asks for powerful tools that drastically reduce the large background while selecting signal events efficiently. Actually such tools are widely needed and used in all parts of particle physics. Regarding the huge amount of data that will be produced at the LHC, the process of training as well as the process of applying these tools to data, must be time efficient. Such tools can be multivariate analysis – also called data mining – tools. In this contribution we present the results for the application of the multivariate analysis, rule growing algorithm RIPPER on a problem of particle selection. It turns out that the meta-methods bagging and cost-sensitivity are essential for the quality of the outcome. The results are compared to other multivariate analysis techniques.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7, 2008
Erice, Italy

*Speaker.

1. Introduction

Multivariate analysis has successfully been employed in many high energy physics data analyses, see, *e.g.*, [Aba08, Af08, Af09]. In high energy physics multivariate analysis is used typically in the following way. A classifier, *e.g.*, a neural network or a decision tree, is trained on Monte Carlo data, the training set. The result is a classifier model that is validated with an independent Monte Carlo data set (the test set) and then applied to real data. The classifier output is a discriminant variable, for example a probability for each candidate to be signal. Cutting on the discriminant variable then allows to optimize the signal yield with respect to the background. Of particular interest is the common case that the background dominates the signal. In the data mining community (in computer science multivariate analysis is also called "data mining"), such problems where there are, *e.g.*, many more background than signal events, are referred to as "imbalanced problems". The class imbalance problem in general corresponds to the problem encountered by supervised classification[1] on domains for which one class is represented by a large number of instances while the other is represented by only a few. A comprehensive review on this problem can, *e.g.*, be found in [Wei04].

Our goal in this contribution is to investigate the question whether the procedure of cutting on the probability described above is the optimal way to deal with an imbalanced problem, or if the solutions used and discussed in the data mining community can improve our results.

2. Classification methods

There are at least four known possibilities for solving imbalanced data problems: The choice of an appropriate classifier [Wei04], the use of the cost-sensitive approach [Wei04, TSK05, WF05], the use of the sampling based approach [Wei04, TSK05] and bagging.

2.1 Choice of an appropriate classifier

The RIPPER [Coh95, TSK05, WF05] algorithm was chosen as the basic algorithm for our analysis. Rule based classifiers are a technique of classifying instances using a collection of "if…then…" rules. For example:

```
(IPpi >= 1.039316) and (DoCA <= 0.307358) and (IP <= 0.270767) and
(IPp >= 0.800645) => class=Lambda

(IPpi >= 0.637403) and (DoCA <= 0.159043) and (IP <= 0.12081) and
(ptpi >= 149.2332) and (IP>= 0.003371) => class=Lambda

=> class=BG
```

Here the variables are arbitrarily called `IPpi, DoCA, IP, IPp` and `ptpi`. The first two lines give *conditions* joined by a logical `and` followed by a decision. This means that if all the *conditions* are true, the instance is predicted to be of class `Lambda`. If not, the next *rule* in this

[1] The occurrence of rare objects can also be a problem in unsupervised learning, as discussed, *e.g.*, in [Wei04].

		PREDICTED CLASS	
		+	−
ACTUAL CLASS	+	TP	FN
	−	FP	TN

Table 1: The confusion matrix for a two class problem. For the meaning of the abbreviations please refer to the text.

rule set is used. In the end if none of the *rules* applies, there is the empty *rule* (the last line) stating that the instance is predicted to be of class `BG`.

The algorithm therefore can classify very fast. In addition RIPPER is also fast in the learning phase, so online tuning can be done. It is known that RIPPER is working well in case of imbalanced data problems [TSK05, Wei04]. A physical interpretation of the rules can be done.

The way RIPPER produces such a rule set is described very shortly in the following. Please refer to the references for a detailed description. These are the main steps:

1. Divide the training set into two sets, one for growing the rules and one for pruning them.
2. Grow a *rule* by adding *conditions* greedily. Remove the instances that apply to this *rule*.
3. Prune the *rule*, *i.e.*, remove less powerful *conditions* to reduce the complexity of the rule.
4. Go to 2. The stopping criteria are based on description length (number of bits needed to encode the rule, see, *e.g.*, [TSK05]) and error rate.
5. Do some optimization by iteration.

2.2 The Sampling based approach

The sampling based approach is widely used for dealing with imbalanced data problems. The main idea of this approach is to modify the number of instances so that the rare class has a better relative representation in the training sample.

Undersampling drops a number of the non-rare events. This method has the potential problem that some useful negative examples may not be present in the training set, so the classifier model will not be optimal.

In case of oversampling the instances of the rare class are replicated until the training set has an equal number of instances of positive class and negative class. The main problem of this method is that it could lead to overfitting for noisy data, because noise events will gain in weight when replicated. Also the time for the creation of the classifier model increases.

2.3 The cost-sensitive approach

The positive class is usually used for denoting the rare class. The following terminology is used for the elements of the confusion matrix (see also Table 1). True positive (TP) is the number of positive instances predicted correctly, false negative (FN) is the number of positive instances wrongly predicted as negative, false positive (FP) is the number of negative instances wrongly

		PREDICTED CLASS	
		+	−
ACTUAL CLASS	+	$C(+,+)$	$C(+,-)$
	−	$C(-,+)$	$C(-,-)$

Table 2: The cost matrix. For the meaning of the abbreviations please refer to the text.

predicted as positive and true negative (TN) is the number of negative instances predicted correctly. The False Positive Rate is $FPR = \frac{FP}{FP+TN}$ (also known as background efficiency), the True Positive Rate is $TPR = \frac{TP}{TP+FN}$ (also called signal efficiency). Varying a parameter of the classifier, the two dimensional plot FPR versus TPR defines the so-called Receiver Operation Characteristic (ROC) curve, which allows to assess the quality of the chosen classifier.

The cost-sensitive approach introduces the so-called cost matrix that encodes penalties for misclassification (and possibly also for correct classification). In Table 2 a cost matrix is shown. Here $C(i, j)$ denotes the cost of predicting an instance from class i as class j. In the cost-sensitive approach to compare the performance of classifier models the overall cost

$$C_{\text{Ovrl}} = TP \cdot C(+,+) + FP \cdot C(-,+) + FN \cdot C(+,-) + TN \cdot C(-,-) \tag{2.1}$$

is used. There are mainly two ways to make an already existing classifier cost-sensitive: threshold adjusting and instance weighting.

To illustrate the first approach, let us denote the fraction of training instances that satisfy some rule t by $p(i|t)$. Typically for a binary classification problem the positive class is assigned to rule t if

$$\begin{aligned} & p(+|t) > p(-|t) \\ \Rightarrow\ & p(+|t) > (1 - p(+|t)) \\ \Rightarrow\ & p(+|t) > 0.5. \end{aligned} \tag{2.2}$$

A cost-sensitive classifier assigns the class label i to rule t if i corresponds to the smallest of the costs

$$C(i|t) = \sum_j p(j|t) C(j,i). \tag{2.3}$$

This means that in the case of, *e.g.*, $C(+,+) = C(-,-) = 0$, the positive class is assigned to a rule t if

$$\begin{aligned} C(-|t) &> C(+|t) \\ p(+|t)C(+,-) &> p(-|t)C(-,+) \\ \Rightarrow p(+|t) &> \frac{C(-,+)}{C(-,+)+C(+,-)}. \end{aligned} \tag{2.4}$$

So the threshold must be modified from 0.5 to $\frac{C(-,+)}{C(-,+)+C(+,-)}$ to obtain a cost-sensitive classifier. This is called threshold adjusting and is equivalent in this case to a cut on the probability described in Section 1. It is also very similar to sampling (see Section 2.2) as the probabilities get weights according to the enhancement of their class corresponding to $C(+,-)$ and $C(-,+)$ in Equation (2.4).

Nevertheless please mind the differences between threshold adjusting and sampling or instance weighting discussed in the following.

The second approach of making a classifier cost-sensitive is to give instances a weight according to the misclassification cost, which is equivalent to sampling in the case of only off-diagonal entries in the cost matrix. These weights are taken into account when the classifier model is created in a way that avoids errors of the more costly type, resulting in a lower overall cost. In the case of some classifiers such as neural networks, the classifier models and thus the performance is not much different from that of threshold adjusting. In other classifiers such as decision trees or rule learners, the new cost matrix produces a different classifier model [Tin02, WF05], thus changing the probabilities in Equation (2.4). The reason is that in many cases the model building uses the error rate to decide on the rules or tree branches, and the error rate depends on the signal to background ratio in the sample. Instance weighting gives more affective and simple models in those cases. A detailed comparison of the two approaches was done in [Zha08].

2.4 Bagging

Bagging (bootstrap aggregation) [Bre96] is a technique that produces new samples from a given data set by uniform random drawing with replacement of instances from the original data set. For each such set a classifier model is created. A test instance is assigned to the class that receives the highest number of votes, *i.e.*, it is classified according to the majority of the decisions of the classifier models. If a probability is needed, the probabilities of the classifier models are averaged.

Bagging works well for unstable classifiers like rule learners or decision trees. Unstable means that the models of these classifiers are prone to change with noise in the data. As bagging leads to some kind of averaging over bootstrap samples, it reduces the noise and thus reduces this effect [Bre96]. It helps unstable classifiers on unbalanced problems because noise affects the rare class more than the common class [Wei04] and bagging reduces the effect of noise. Thus bagging also reduces overfitting.

3. Lambda hyperon selection algorithm

A sample of $7.7 \cdot 10^5$ LHCb minimum bias Monte Carlo events has been used to select Lambda candidates in the decay $\Lambda \to p^+\pi^-$. Candidates where taken as combinations of two track pairs with constraints on its invariant mass and the distance of closest approach.

For the creation of the selection algorithms and meta-methods the data mining package WEKA [WF05, WF] has been used in version 3.5.7. For the main selection the following set of ten geometric and kinematic variables has been chosen.

- $DoCA$ – distance of closest approach
- FL – signed flight-length (positive if the decay is downstream of the primary vertex, negative otherwise)
- $c \cdot t$ – flight-length in Λ frame
- IPp – impact parameter of the proton at the primary vertex

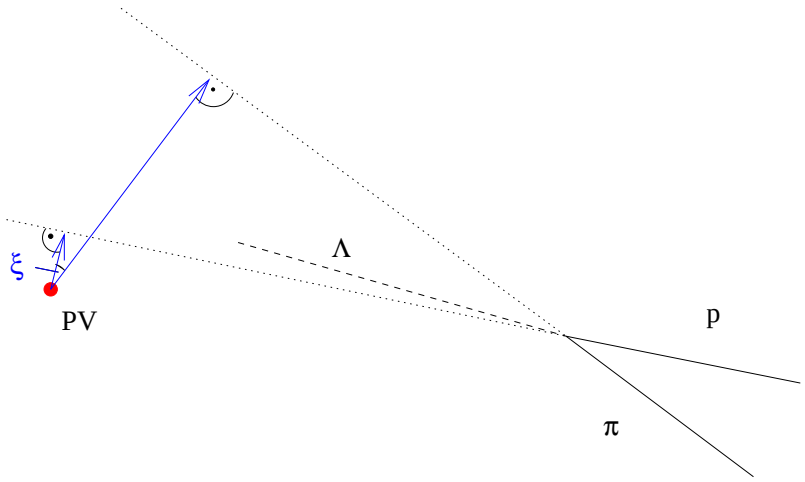

Figure 1: Definition of the angle ξ. PV is the primary vertex, the blue arrows are the impact parameter vectors for the proton and the pion track respectively.

- $IPpi$ – impact parameter of the pion at the primary vertex
- $v_2 = \ln(\frac{IPpi^2+IPp^2}{IP^2})$
- ptp – transverse momentum of the proton
- $ptpi$ – transverse momentum of the pion
- $\tan\vartheta = \frac{pt}{pz}$ of the Λ
- $\cos\xi$ – angle between the impact parameter vectors.

The definition of ξ is illustrated in Figure 1.

For the selection of Λ hyperons a two step procedure was applied. In a preselection step a classifier model is constructed that efficiently reduces the amount of background without loosing too many signal events. In this step a cost matrix with zero diagonal elements and $C(\Lambda, BG) = 1$, $C(BG, \Lambda) = 100$ has been used. The number of bagging samples used for the preselection was 10. For the main selection a cost matrix with zero diagonal elements and $C(BG, \Lambda) = 1$ has been used. The instance weighting cost $C(\Lambda, BG)$ has been scanned from 10 up to 200 in steps of 10 creating a new set of rule sets in each step (see Section 2.3). Here the number of bagging samples was equal to 25. Figure 2 shows the resulting ROC curve. We use the ROC curve since it is independent of the ratio between signal and background in the test set. Please mind that in this representation a classifier is better if the curve is more to the left top of the graph. Since we use bagging, the scatter of the ROC curve gives an impression of how the result depends on the choice of the training sample. The invariant mass plots are presented in Figure 3.

In Figure 4 the ROC curves for RIPPER without bagging and instance weighting and that one with the application of these meta-methods is shown. The latter shows a much better result.

To compare different algorithms the ROC curves for three different classifiers using WEKA v3-5-7 have been produced: the decision tree algorithm C4 [Qui93, WF05], a multilayer perceptron and the rule based algorithm RIPPER. RIPPER was used with its default settings as proposed

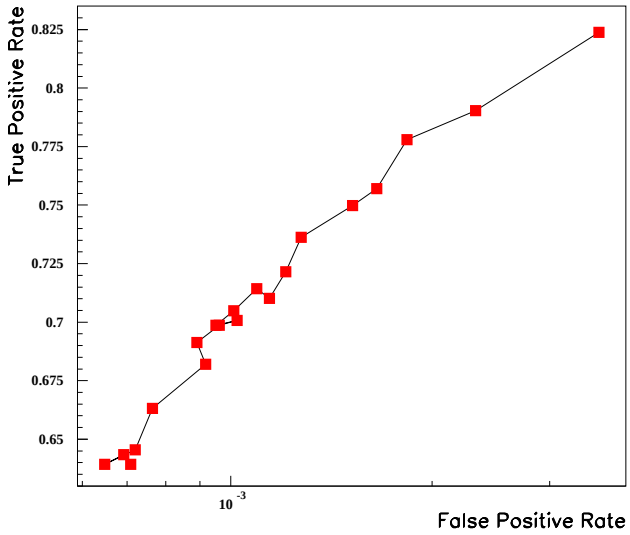

Figure 2: ROC curve for our Lambda hyperon selection. Note that each point in this plot is the result of a different set of rule sets.

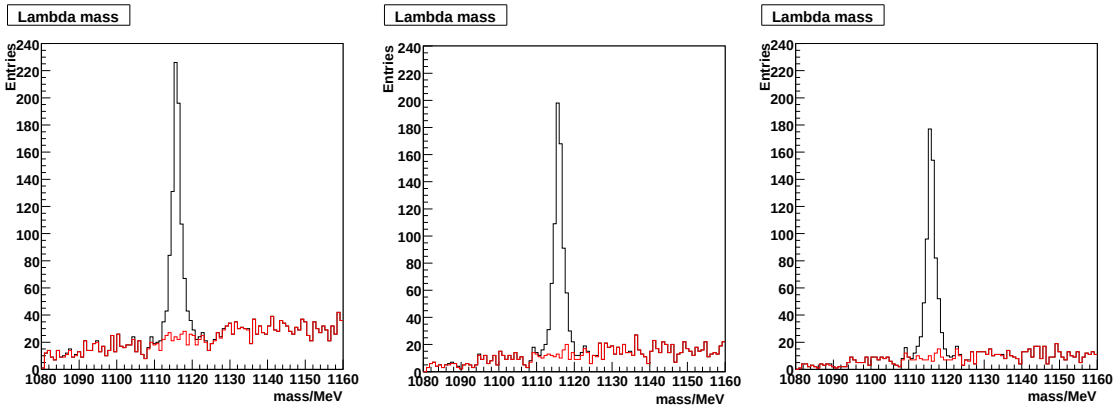

Figure 3: Lambda mass plots for cost on FN equal to 30, 100 and 200 (left to right). The black line represents all accepted candidates while the red line represents only the accepted background candidates.

in [Coh95]. For C4 the standard settings of WEKA have been used which include pruning. The multilayer perceptron with backpropagation consisted of 3 layers, 6 internal nodes and a binary output; the input variables have been normalized to lie between -1 and 1. All parameters have been taken to be the same as in the base version of the analysis, the preselection step of RIPPER was skipped to have a better comparability. The ROC curves in Figure 5 show that RIPPER gives the best result for almost all values of FPR. It is also the fastest algorithm among these three.

The two approaches instance weighting and threshold adjusting have been applied to the Lambda selection with RIPPER for the same parameters and meta-methods. In Figure 6 the two corresponding ROC curves are shown. By increasing the cost ratio the threshold adjusting method could not be improved below an $FPR \sim 2 \cdot 10^{-3}$ because FPR and TPR become equal to 0. In addition this approach is slower than instance weighting when the cost is fixed and gives much more complex rule sets.

In Figure 7 we give three ROC curves for different numbers of bagging iterations. The result

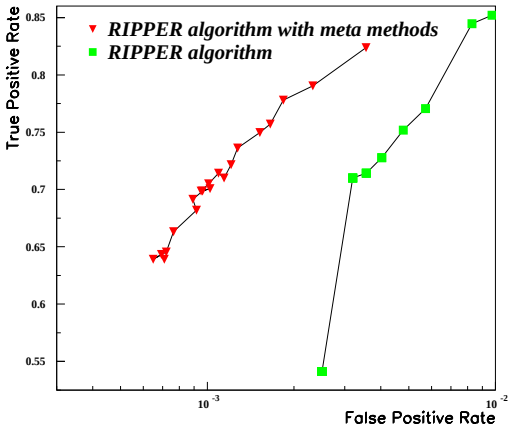

Figure 4: ROC curves for the classifier without the application of bagging and instance weighting and with the application of these meta-methods.

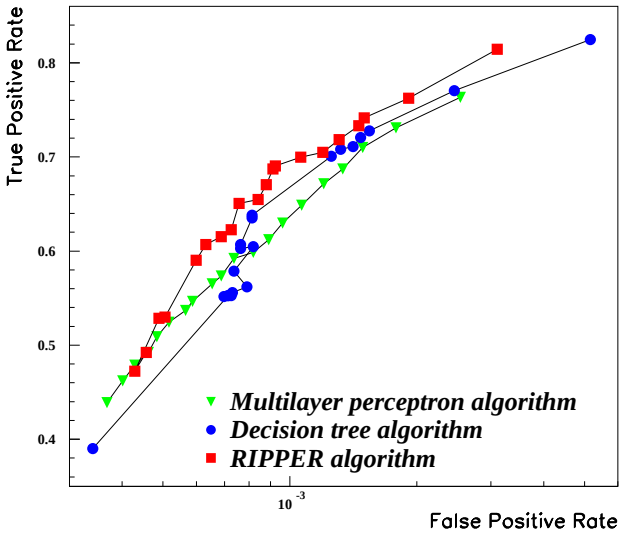

Figure 5: ROC curves for the three different base classifiers.

for a number of iterations of 25 is practically indistinguishable from that with 40 iterations. So 25 bagging iteration seem to be enough. Using only three iterations on the other hand gives a much worse result which is also rather noisy.

The TMVA package [HSS+07] for multivariate analysis is included in the ROOT framework [BR]. The best result obtained with the WEKA package is compared to the best obtained with TMVA v3.9.4 [Vos08], a boosted decision tree with pruning. In Figure 8 the corresponding ROC curves are shown. Our WEKA results are significantly better than those of TMVA.

4. Conclusion

Methods of supervised classification can be used for particle selection in experimental particle

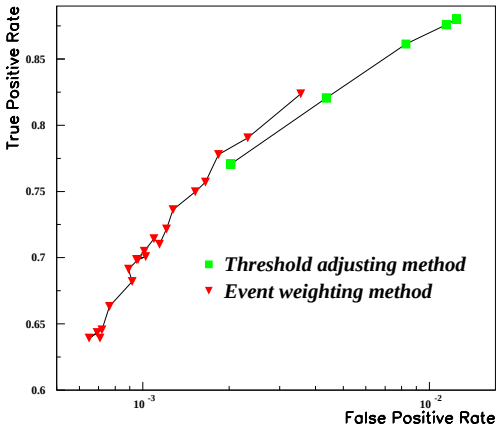

Figure 6: ROC curves for the instance weighting and threshold adjusting approach for making classifiers cost sensitive.

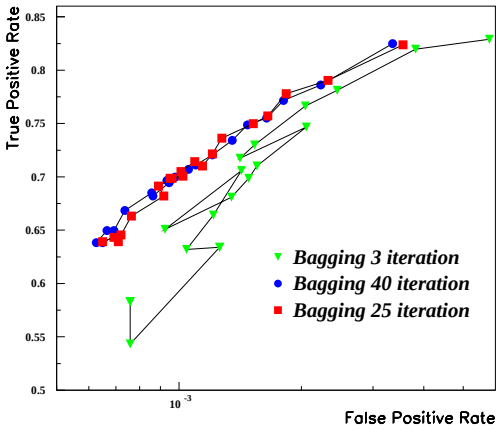

Figure 7: ROC curves for different number of bagging iterations.

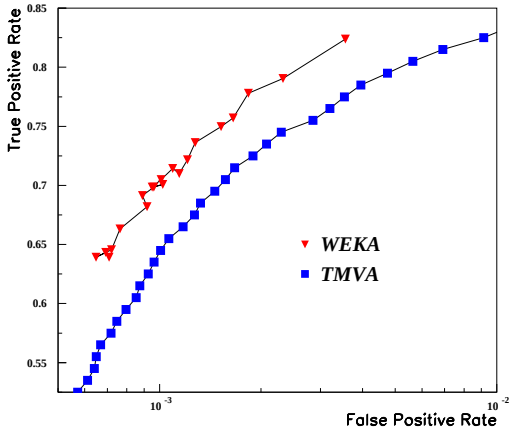

Figure 8: ROC curves for different packages of programs, TMVA results from [Vos08].

physics. Typically we have to deal with what is called imbalanced problem in the data mining context, *i.e.*, much more background than signal instances. We have shown that there are more sophisticated methods to deal with these kinds of problems than simple cuts on the probabilities. In our sample analysis we have combined the imbalanced problem proof rule learner RIPPER with bagging and instance weighting to make the classifier cost-sensitive. We have shown that this gives very good results, mostly better than using other base classifiers or the package TMVA. Especially bagging and instance weighting are found to be very important ingredients.

References

[Aba08] V. M. Abazov, *et al.* Search for the Standard Model Higgs Boson in the Missing Energy and Acoplanar b-Jet Topology at s=1.96TeV. *Physical Review Letters*, 101(25):251802–+, December 2008.

[Af08] B. Aubert and for the BABAR Collaboration. Search for the decay Bˆ+ –> K_Sˆ0 K_Sˆ0 piˆ+. *ArXiv e-prints*, November 2008.

[Af09] B. Aubert and for the BABAR Collaboration. Improved Measurement of $Bˆ+\to\rhoˆ+\rhoˆ0$ and Determination of the CKM Angle α. *ArXiv e-prints*, January 2009.

[BR] R. Brun and F. Rademakers. Web: `http://root.cern.ch`.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Coh95] W. W. Cohen. Fast effective rule induction. In Machine Learning: Proc. of the 12th International Conference on Machine Learning. Morgan Kaufman, 1995.

[HSS+07] A. Hocker, P. Speckmayer, J. Stelzer, F. Tegenfeldt, H. Voss, K. Voss, A. Christov, S. Henrot-Versille, M. Jachowski, A. Krasznahorkay, Jr., Y. Mahalalel, R. Ospanov, X. Prudent, M. Wolter, and A. Zemla. TMVA - Toolkit for Multivariate Data Analysis. *arXiv:physics/0703039*, March 2007. Web: `http://tmva.sourceforge.net`.

[Qui93] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[Tin02] K. M. Ting. An Instance-Weighting Method to Induce Cost-Sensitive Trees. *IEEE Trans. on Knowledge and Data Engineering*, 14(3):659–665, 2002.

[TSK05] P.-N. Tan, M. Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[Vos08] H. Voss, 2008. Private communication.

[Wei04] G. M. Weiss. Mining with rarity: A unifying framework. *Sigkdd Explorations*, 6(1):7–19, 2004.

[WF] I. H. Witten and E. Frank. Web: `http://www.cs.waikato.ac.nz/ml/weka`.

[WF05] I. H. Witten and E. Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufman Publishers, 2005.

[Zha08] H. Zhao. Instance Weighting versus Threshold Adjusting for Cost-sensitive Classification. *Knowledge and Information Systems*, 15(3):321–334, 2008.